Validation (not just Verification) of Deep Space Missions[1]

Riley M. Duren
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099
818-354-5753
Riley.M.Duren@jpl.nasa.gov

Abstract – Verification & Validation (V&V) is a widely recognized and critical systems engineering function. However, the often used definition "Verification proves the design is right; validation proves it is the right design" is rather vague. And while Verification is a reasonably well standardized systems engineering process, Validation is a far more abstract concept and the rigor and scope applied to it varies widely between organizations and individuals. This is reflected in the findings in recent Mishap Reports for several NASA missions, in which shortfalls in Validation (not just Verification) were cited as root- or contributing-factors in catastrophic mission loss. Furthermore, although there is strong agreement in the community that Test is the preferred method for V&V, many people equate "V&V" with "Test", such that Analysis and Modeling aren't given comparable attention. Another strong motivator is a realization that the rapid growth in complexity of deep-space missions (particularly Planetary Landers and Space Observatories given their inherent unknowns) is placing greater demands on systems engineers to "get it right" with Validation.

TABLE OF CONTENTS

1. MOTIVATIONS

Increasing complexity of the missions being launched represents an important driver on the rigor employed in the Systems Engineering disciplines of Validation & Verification (V&V). The upcoming suite of deep-space missions requires dramatic new technologies in many cases. In the space-astronomy arena, missions are driving the state of the art in many areas: Kepler's few parts-per-million photometry, the Space Interferometry Mission (SIM)'s few micro-arcsecond astrometry, the Laser Interferometric Space Antenna (LISA)'s fempto-G acceleration sensing, and the Terrestrial Planet Finder (TPF)'s ultra-high contrast (10^{-10}) imaging needs are all setting new targets for performance. Getting new technologies such as pico-meter metrology, sub-angstrom optical wavefront control, and separated-spacecraft, cryogenic nulling interferometry to work on Earth is difficult enough. The Validation challenges associated with converting these to reliable, space-borne systems are formidable.

Likewise, as the scope of NASA's Solar System Exploration Program expands, future missions to Mars and the outer planets drive the need for highly autonomous spacecraft remote-agents, real-time hazard sensing and avoidance in planetary landers, advanced propulsion and energy systems, and the ability to operate in extremely high radiation environments. Many of the above capabilities are cost-prohibitive or impractical to test in an end-to-end fashion prior to launch, thus placing an increased burden on modeling and simulation as part of a robust Validation program. There is also a caution here: projects faced with daunting technological challenges often succumb to tunnel vision and focus on "invention" while neglecting the more mundane aspects of the system, such as the spacecraft bus. Flight in deep-space is still far from routine. Projects must continue to apply significant attention to basic health and safety issues, such as fault tolerance and fault protection design and validation. [Note: while this paper focuses on Deep-Space missions, these concepts should be applicable to other types of space missions and complex systems in general].

Another motivation (and the central thesis here) is the specific role of Validation as distinct from Verification. Some observations follow. <u>Verification</u> is a well-established SE process. However, in the author's experience, <u>Validation</u> is a fuzzy, organic, poorly-understood concept (or at the very least, "your mileage may vary" widely between projects and organizations). Another observation: test is the preferred method for Validation & Verification. As a result, many system

[1] [1] 0-7803-8155-6/04/$17.00 © 2006 IEEE

Mission/year	Mishap	Validation-related Contributing Factors
Genesis/2004	G-switch installed backward → parachute not deployed → hard landing	"no system-level test" (of G-switch)
Columbia/2003	debris damaged thermal tiles → loss of crew and vehicle	"current tools, including the Crater model, are inadequate …" "flight configuration was validated using extrapolated test data … rather than direct testing" [1]
Comet Nucleus Tour (CONTOUR)/2002	overheating of s/c by solid rocket motor plume → vehicle lost	"Project reliance on analysis by similarity"[2]
Wide-field InfraRed Explorer (WIRE)/1999	electronics startup transient → early cover jettison → cryogen boil-off → science mission lost	"failure to correctly identify the source of the signal which caused the Electro Explosive Device (EED) Simulator to "latch" upon Pyro Box power-up during spacecraft integration testing."[3]
Mars Polar Lander (MPL)/1998	software flaw → descent engine shut-off too soon → vehicle lost	"employed analysis as a substitute for test in the verification and validation of total system performance...tests employed to develop or validate the constituent models were not of an adequate fidelity" [4]

Table 1 – Excerpts from recent NASA Mishap Reports

engineers equate V&V with Test – to the extent that Modeling & Analysis aren't given comparable attention. For example, the concept of Test As You Fly (TAYF) is broadly recognized in the aerospace community. However, we have no global recognition of a similar concept for Model As You Fly (MAYF). With growing mission complexity the need to identify and validate mission-critical models is becoming increasingly important. And a final motivating observation about Validation – we must learn to recognize that no Model or Test is perfect – hence the system engineer must rigorously establish Model Uncertainty Factors (MUFs) and Test Uncertainty Factors (TUFs).

2. SOME DIFFICULT LESSONS

Recent attempts to implement missions with the "faster, better, cheaper" (FBC) approach have suffered from mixed results. On the one hand, the Mars Pathfinder, Lunar Prospector, Near Earth Asteroid Rendezvous (NEAR), Deep Space 1 (DS1), and Deep Impact missions were successfully implemented in the FBC mode. However, those successes have been tempered by catastrophic failures of the Mars Climate Orbiter (MCO), Mars Polar Lander (MPL), Wide-field Infrared Explorer (WIRE), and Comet Nucleus Tour (CONTOUR) missions and Genesis hard-landing. This mixed track record encourages continued refinement of our project implementation practices to emphasize "success first", and in particular, the validation techniques needed to ensure system robustness critical on missions with relatively small teams in which every "link" in the risk-avoidance chain must be protected. One should also remember the lessons of the Hubble Space Telescope (HST), Mars Observer (MO), and Columbia, which demonstrated that greater resources do not always correlate to reduced risk. Lessons-learned reports for such missions repeatedly highlight the paramount importance of a thorough end-to-end V&V program when trying to balance risk and cost. Validation techniques, which are often less consistently applied than Verification activities, hence result in some of the major findings in various mishap reports, as shown in Table 1.

3. CONTEXT & TERMINOLOGY

While Independent Validation & Verification (IV&V) of *software* has received considerable attention recently, standards for the broader *mission-level* V&V activity are rather fuzzy (despite attempts by ISO and other organizations), other than the mundane aspects of verification. Likewise, the scale of mission-level V&V efforts and basic approach varies widely from project-to-project. In fact, there appears to be a *cultural* aspect of how validation is treated in various organizations. Namely, "great system houses" are often adept at using validation to catch problems in the design phase whereas "great integration houses" have demonstrated an ability to find creative solutions for horrendous problems that arise late in the implementation phase[5]. Missions have been successfully implemented by both cultures but the latter typically suffers from increased risk and higher costs (it's cheaper to correct problems earlier).

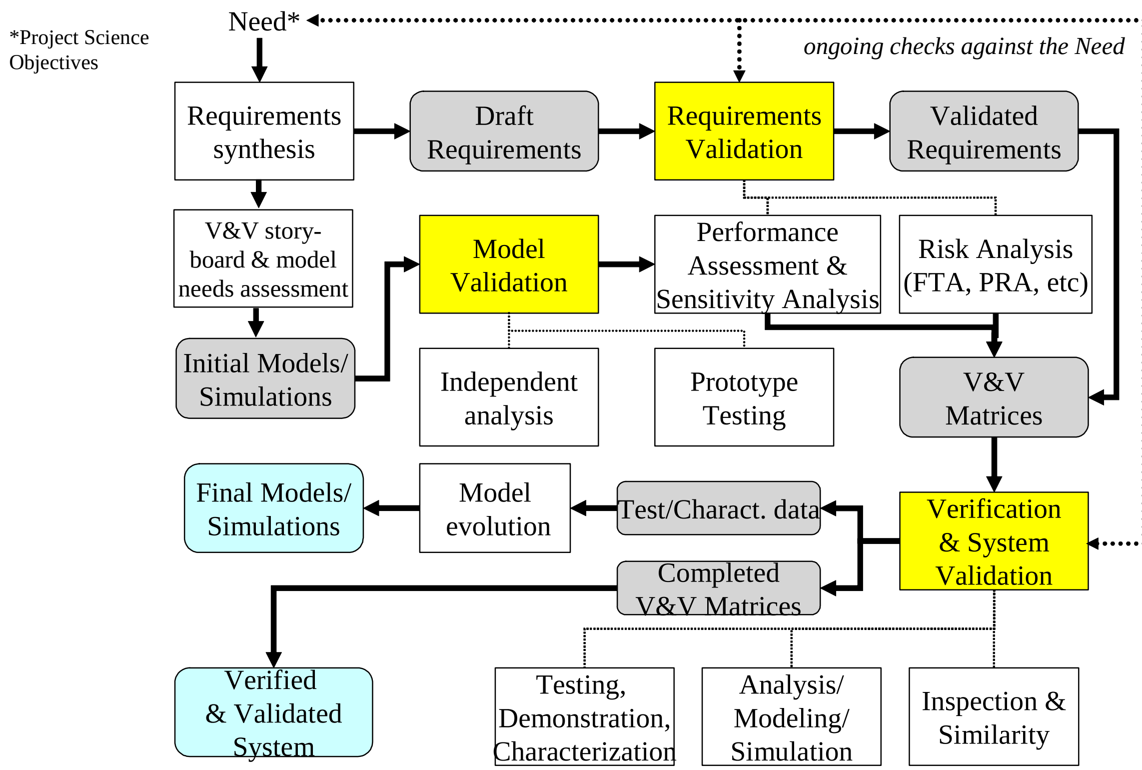


Figure 1 – Generic V&V Program Structure

The following definitions are provided to illustrate the subtle but important distinctions between different aspects of a V&V program. The commonly-used phrase "Verification proves the design is right; validation proves it is the right design" is correct but not very specific[6]. The author has used a combination of sources including the NASA Systems Engineering Handbook, ISO 9000 Handbook, and personal experience to refine the common definitions as follows[7]. **Verification** is proof of compliance of the as-delivered system with *specific* requirements (i.e., "does what we built meet the requirements we wrote?"). While proving compliance with top-level performance requirements can sometimes be a challenging task, the basic concepts behind verification are fairly straightforward and universally recognized, such as the use of verification matrices. Validation is a more nebulous concept and can be broken into three sub-definitions: requirements validation, model validation, and system validation. **Requirements Validation** is proof that the requirements (and hence the system design) should satisfy the customer's *Need* or purpose before the system is actually built. **Model Validation** is proof that the models and simulations to be used for requirements and system validation are correct. **System Validation** is proof that the as-delivered system (all project elements operating end-to-end in the expected flight environment with reasonable stressing conditions) will meet the driving Need (i.e., "does what we built meet the objectives?"). It is important to recognize these distinctions between Verification and Validation.

The structure of a generic V&V program is shown in Figure 1. The driving *Need* is used to set the initial requirements and is frequently referred back to as part of the validation activity. Initial model validation is done to ensure the models/simulation can be used safely to support requirements validation. In addition to performance analysis/modeling/ simulation, requirements validation includes *risk analysis* such as Fault Tree Analyses and Probabilistic Risk Analyses to ensure the design will be robust against failures or off-nominal situations. This will be discussed further in upcoming sections. V&V requirements and a set of tracking matrices are used to cover requirements validation, model validation, verification, and system validation.

System-level V&V, including roll-up from lower-level V&V, consist of testing and calibration (characterization) of the as-built system, analysis, modeling, and simulation of un-testable aspects, and inspection for others. The output of the V&V activity includes updates to the V&V matrices and test/calibration data for updating the system models and simulations (for potential future use in flight). When the V&V matrices are complete, the system can finally be considered validated (should satisfy the Need). Of course, a system isn't truly 100% validated until the mission is successfully operational but every project must strive to achieve a reasonable confidence level before launch. The definition of reasonable varies from project to project although most strive for > 90% confidence at launch.

Project Systems are (at least) *three-dimensional* in nature as illustrated in Figure 2. Therefore, the V&V programs applied to them (and corresponding thought processes) must also be multi-dimensional. The key concepts behind this figure are as follows. First, to understand Validation, the system engineer must fully grasp the context in which the Project System “lives”. For most NASA missions, the goal is to obtain new Scientific Knowledge about some physical Phenomenon Under Study. The Project System is that collection of “stuff” (hardware, software, people, processes, and facilities) that is created to operate in some environment(s) and provide the necessary bridge between the Phenomenon and Knowledge. The system engineer must be capable of visualizing the Project System’s “transfer function” in order to successfully validate it. Second, because in the real-world projects exist as complex, multi-dimensional constructs rather than simple, single-dimensional projections, one cannot really prove that it will work as *Needed* if Validation is only performed in isolated dimensions. Before moving on to discuss each dimension, there are two complications worth highlighting:

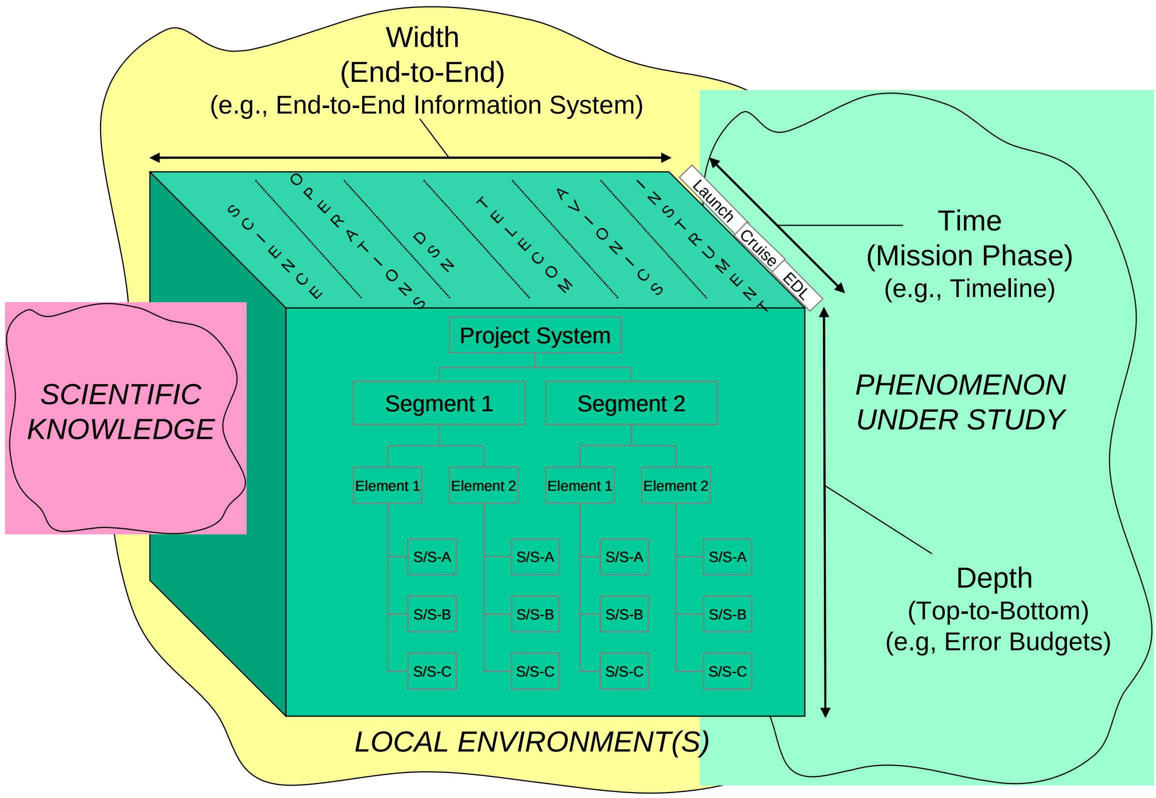


Figure 2 – Project System Context & Multi-Dimensional Nature

(1) The dimensions are not entirely orthogonal so we must cognizant of where significant cross-coupling exists.

(2) Accurately perceiving the 3D-picture by gestalt is very difficult – instead, this requires multiple “looks” from different perspectives.

There is an end-to-end or front-to-back aspect of any mission that spans its entire functional and performance space, which can be thought of dimensionally as *width*. The End-to-End Information System (EEIS) of a deep space mission is a good example of project width. The EEIS includes the acquisition of data (observables) from science instruments, through formatting and downlink by spacecraft avionics (command & data handling subsystem) and telecommunication subsystem, delivery via the Deep Space Network (DSN), and processing by the Ground Data System and subsequent Science Analysis. The EEIS also includes the return path of mission planning and uplink commanding.

The second dimension of a project is the top-to-bottom aspect or *depth*. This is the realm of the performance error-budget and functional decomposition. The risks inherent with working in this dimension are described in more detail in the upcoming section on Requirements Validation. The hierarchical structure of the project in the depth dimension is also illustrated by the familiar systems engineering "V-model" (Figure 3) which shows the process by which requirements are sub-allocated and validated on the downstroke and then the design and as-built system are verified and validated on the upstroke[5]. Whereas verification is typically performed at all levels, validation in its most rigorous and comprehensive form occurs at the system level. However, in a well-executed project, some validation is performed at all levels. For example, a designer of an electronic board will typically perform a worst-case analysis to assess the design robustness to stressing conditions. Such lower-level validation activities are flowed upwards to the next level, playing a part in overall system validation.

The third dimension of a project is *time*. In this context, time has (at least) two distinct implications. First, when planning validation activities one must carefully consider the functional evolution of the mission over time. For example, a Mars lander has distinct mission-critical functionality unique to each mission-phase: Launch, Cruise, and Entry, Descent, & Landing (EDL). This phase-dependent functionality (particularly time-critical events and phase transitions) must be carefully considered by the systems engineer when designing validation activities (i.e., relying on a functional decomposition or depth view alone is not sufficient).

Secondly, the time dimension is important in that the V&V program must *consider* the entire project life-ccycle, from formulation, definition, development, test, operations, and data reduction (phases A through F in NASA parlance) even if the V&V effort itself is carried out primarily during the last year or two before launch. Figure 4 illustrates how the V&V program evolves over the project life cycle. Note: while uplink command validation and software-patch validation are important during operations, they are considered peripheral to this discussion which is focused primarily on ensuring *mission-readiness*.

The important point here is that the project must go through the exercise of drafting a V&V Plan early in definition phase and explore the above topics. Considering and agreeing on the overall risk posture and corresponding scope of the V&V program is essential to achieving the proper balance.

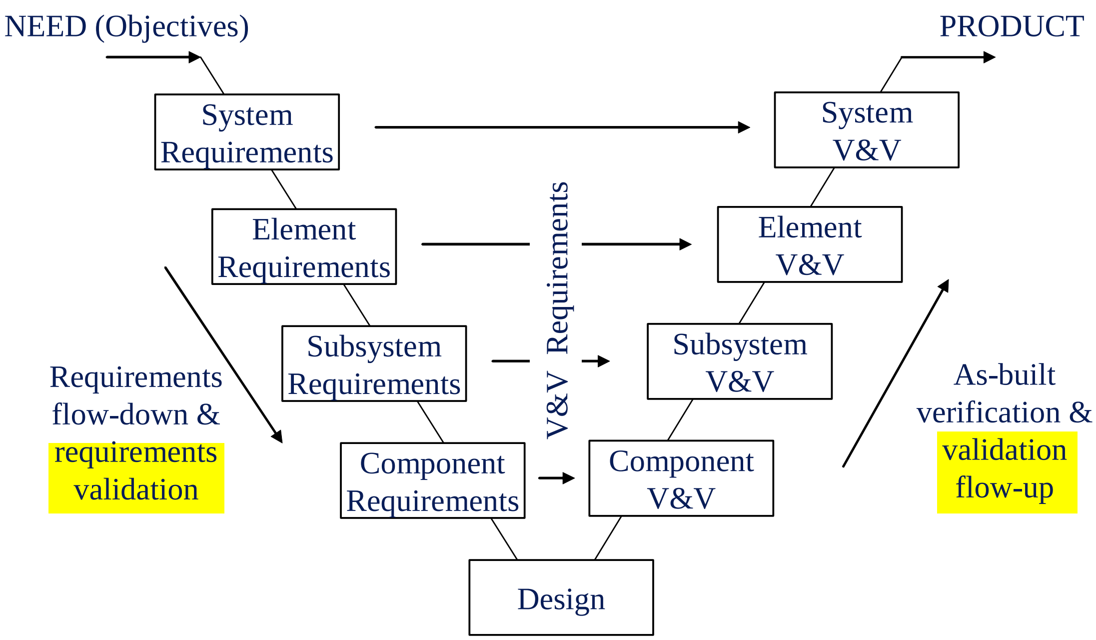


Figure 3 – Systems Engineering V-Model

4. REQUIREMENTS VALIDATION

In proving the system will meet the ultimate Need, validation must address several aspects of the requirements and system design: *correctness*, *completeness*, *achievability*, *verifiability*, and *robustness*. Clearly, the requirements governing the system design must be correct and complete in order to satisfy the Need. The requirements must likewise result in a design that is achievable given the allocated project resources. To avoid unreasonable risk, requirements must be such that they can be verified once the system is built. Finally, the requirements should result in a system that is both robust to variations in performance beyond the nominal operating range as well as robust in the presence of reasonable fault conditions. These points were driven home with lessons-learned in the wake of the Mars Climate Orbiter (MCO) and Mars Polar Lander (MPL) failures, namely: verification such as a Test Like You Fly philosophy can never relieve engineers of their top priority – to get it right the first time[8]. Specific recommendations in this area included:

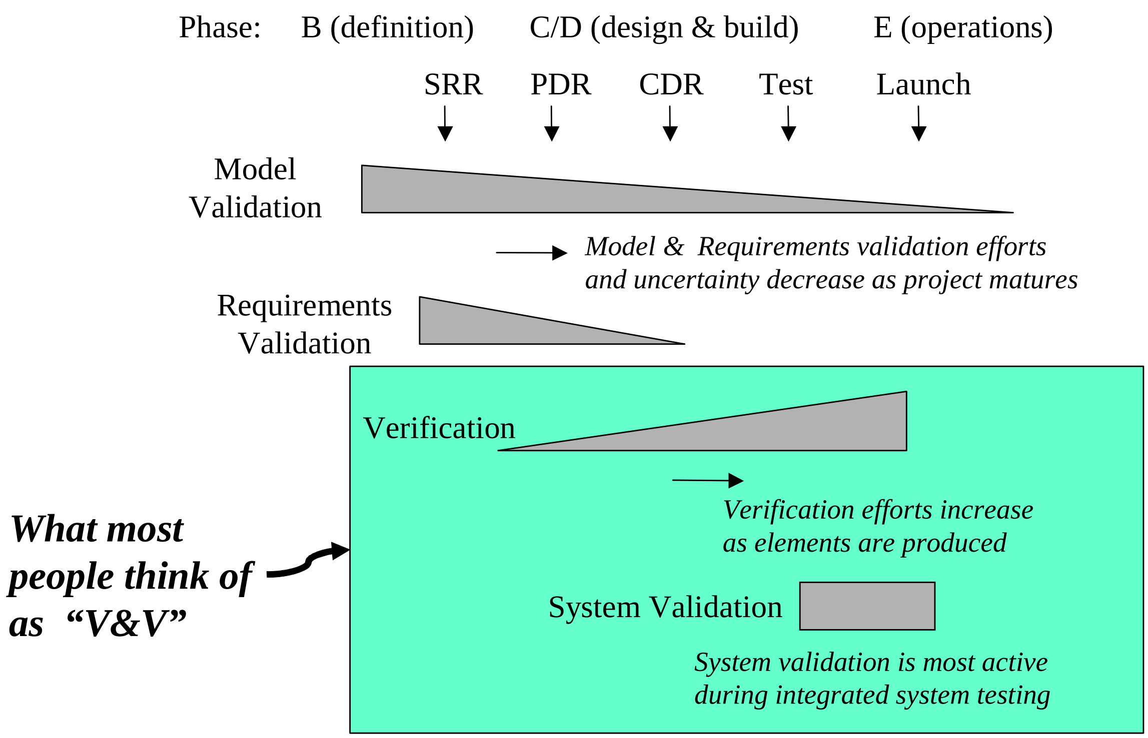
a) Avoid stating requirements in a negative sense ("shall not do X") because they are notoriously difficult to verify
b) Ensure that lower-level (e.g., subsystem) requirements are covered properly by a parent requirement at the system level – since the latter are generally subject to more rigorous, formal V&V and thus more likely to catch problems during later verification.
c) Avoid compound requirements (e.g., "shall do X and Y") as they are often difficult to validate together.

The system engineer should maintain a mental check-list of the following considerations when validating requirements:

- System complexity often defies the "black-box" model
  - Tight coupling/interdependencies
  - Pourous/amorphous interfaces
- Systems are multi-dimensional – "hyper cube" model
  - Interdependencies often cross dimensional boundaries
  - Not always orthogonal
- Requirements exist in one of two basic camps
  - What the system must be
    - Performance
  - How the system should behave
    - Functionality/Operability

Given that Validation is a three dimensional problem it is important to study the driving requirements from multiple vantage points to avoid missing something critical. The following tools and techniques are available for validation, each offering a unique and important viewpoint:

1. Functional flow diagrams
2. Performance Error Budgets
3. State Analysis
4. Performance Sensitivity Analyses/Models
   a. Merit Functions
   b. Monte Carlo Simulations
5. Risk Analyses
   a. Fault Tree Analysis
   b. Probabilistic Risk Analysis
   c. Failure Modes, Effects, and Criticality Analysis
   d. Worst-Case Analysis
6. Early hardware/software testing



*note: while the bulk of the V&V Program exists in Phase C/D for purposes of certification of flight readiness, other Validation efforts are performed during operations

Figure 4 – V&V across the Project Life-cycle

Functional Flow Diagrams

End-to-end system functionality (the "width" dimension of the V&V program) can be studied with functional flow diagrams. The exercise of understanding the logical flow between project elements from front-end to back-end offers insight into the adequacy of interface requirements and overall system utility. In addition to addressing hardware/software aspects, functional flow diagrams are also important for studying *operational* processes and the requirements placed on them, particularly for human-machine interactions and fault response.

Performance Error Budgets

Likewise, comprehensive error budgets are useful in understanding the top-to-bottom flow of performance requirements and capabilities (addressing the depth dimension of V&V). A simplified example of an error budget from the Kepler planet-detection mission is shown in Figure 5. The driving Need/Science Objective and resulting top-level system requirements are highlighted. Error budgets also represent an example of the temporal aspect of validation. Early in the project, engineers use error budgets in a top-down mode to sub-allocate performance requirements to the different elements. Then, as the project progresses and design maturity improves, the error budgets are used in a bottom-up mode to predict the expected performance. Improving the fidelity of the error budgets in this fashion is one goal of the model validation effort.

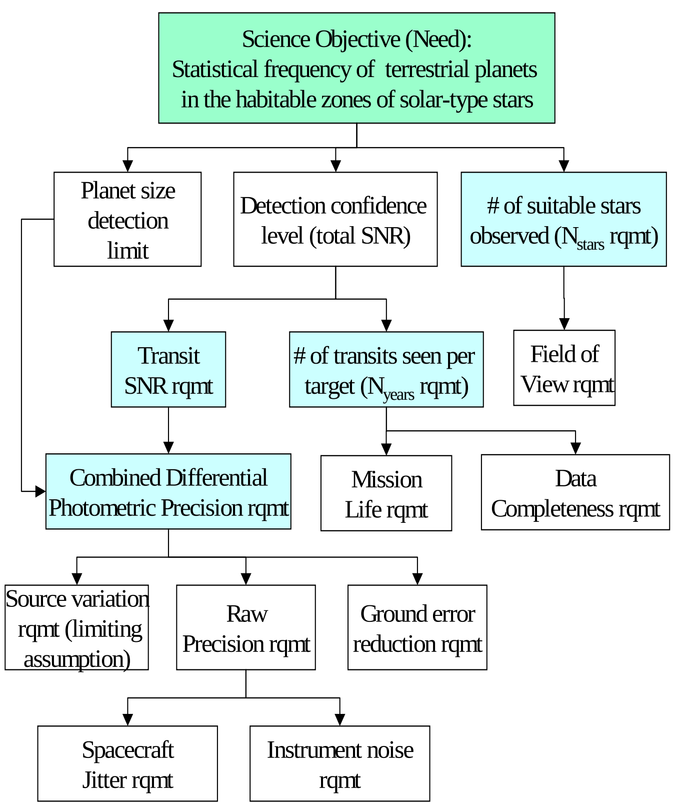


Figure 5 – Performance Error Budget for Kepler

A potential pitfall to watch for in error budgets is the appropriate treatment of systematic errors versus uncorrelated, random noise. This topic is beyond the scope of this paper, but the performance analyst must be scrupulous in understanding and accounting for the effects of systematic errors and the calibration and error rejection techniques employed in data-reduction. Carefully accounting for small errors and residuals are necessary to avoid being either over-conservative or too optimistic in estimating overall system performance. System engineers should demand that performance error budgets explicitly depict all potential error sources (even

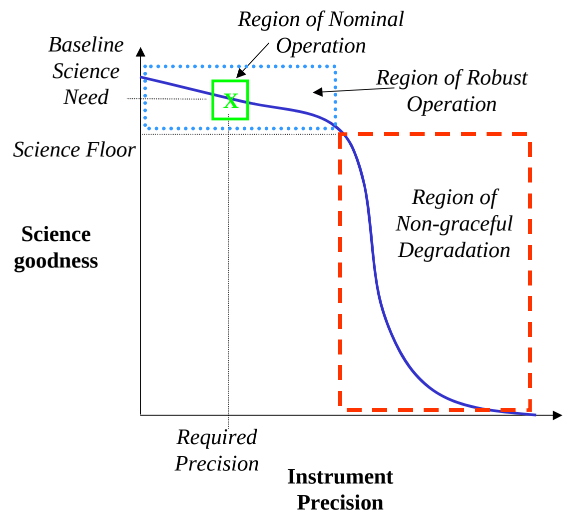


Figure 6 – Science Merit Function (Sensitivity Analysis)

if the magnitude of the error contribution is negligible, the exercise must be completed).

Robustness and Graceful Degradation

Projects sometimes make the mistake of unintentionally creating requirements and designs that result in operation at or near "cliffs". While a system may be designed to meet performance specifications within a fairly tight set of tolerances around a required central value, it may fail precipitously in the event of relatively minor excursions from the region of nominal operation. The concept of *graceful degradation* is key to successfully implementing deep-space systems in the faster, better, cheaper environment. Properly executed validation programs enable graceful degradation by using performance sensitivity analyses and design risk analyses to identify cliffs and soft-spots, thus providing the project sufficient insight to guide risk versus cost (mitigating action) trades.

Performance Sensitivity Analyses

Merit Functions and Monte Carlo simulations are two useful tools for assessing the robustness of a system in terms of its overall performance. Monte Carlo simulations are well-described in the literature and can be very helpful for perturbing system parameters such as spacecraft pointing jitter in stressing cases to assess the impact on top-level performance[6].[5]

Merit Functions are models that provide the system engineer and the customer (such as a Principal Investigator or PI) with a method of studying the sensitivity of the mission Need to changes in key mission parameters. Another way of thinking about this is: *what is the science sensitivity to the mission parameters*?

As an example, consider the Kepler mission. The PI has identified a Need to determine the frequency of terrestrial planets in the habitable zones of solar-type stars. The quality of the Science produced (i.e., the number of appropriate planets found per star observed) is some function of a few key mission parameters: instrument precision, detection signal-to-noise ratio (SNR), # of stars observed, # of years observed. To determine the sensitivity of the Kepler "science goodness" we can establish some Merit Functions that allow us to study crucial "partial derivatives":

$$\frac{\partial(S)}{\partial(SNR)},\ \frac{\partial(S)}{\partial(CDPP)},\ \frac{\partial(S)}{\partial(N_{stars})},\ \frac{\partial(S)}{\partial(N_{years})}$$

where Science goodness S = f(CDPP, SNR, Nstars, Nyears)
CDPP = combined differential photometric precision
SNR = detection signal-to-noise ratio
Nstars = number of stars observed
Nyears = number of years observed

Again, the goal is to deliver a system that is robust to variations in the key mission parameters by remaining in the flat portion of the performance curve. Figure 6 provides another, hypothetical example of a Merit Function in which the sensitivity of the science goodness with respect to instrument precision is plotted. The X marks the intersection of the Baseline Science Need with the Required Precision (the former should have driven the latter during synthesis). The small box around the X represents the Region of Nominal Operation. The Region of Robust Operation is defined by a combination of the Science Floor (minimum mission success criteria) and the edge of the "cliff" which marks the boundary of the Region of Non-Graceful Degradation. The mission should strive to work properly within the Region of Robust Operation.

Note: science floor is the most important driver when setting the Region of Robust Operation. In this idealized example, the design has been cost-optimized such that the science floor intersects the knee in the curve (it is of course acceptable and preferable to have some distance between the two).

<u>Early Hardware/Software Testing</u>

When performance sensitivity analyses and design risk analyses identify a potential cliff or soft spot, it is often prudent to perform early tests using engineering model hardware and/or software testbeds to confirm such predictions and/or assess mitigating designs. Again, such tests are only as good as their design – a well thought-out V&V plan can guide what and how to test during a

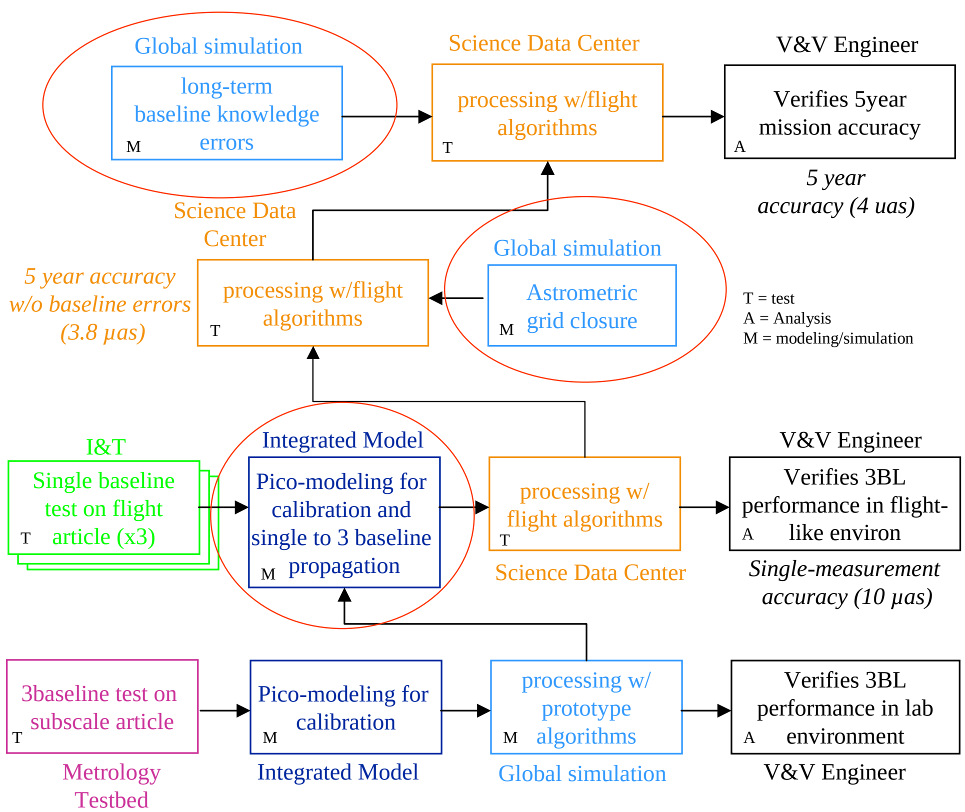


Figure 7 – V&V storyboard for SIM wide-angle astrometric accuracy

project's definition phase.

5. MODEL VALIDATION

Models (including simulations) play important roles both in requirements validation as well as subsequent verification and system validation. The uses of models in requirements validation were described above. For the later phases of V&V, models are often used to "bridge the gaps" in the system verification and validation effort that cannot be directly tested. Some models (or certain aspects of them) should be considered *mission-critical* if errors in such areas could mask problems leading to failures in the operations phase of the project.

In a cost-constrained project, the following minimum approach should be used to identify and validate mission-critical models and simulations:

1) In early drafts of the Project V&V Plan, identify the mission-critical models (create verification and system validation "storyboards" and show where and how models and simulations fit into the overall scheme relative to testing).
2) Establish requirements on mission-critical model functional capabilities and accuracy
3) Validate the mission-critical models in terms of those driving requirements
4) Maintain configuration control of the validated models (treat as mission- critical software)

An example of the V&V storyboard effort is shown in Figure 7. This addresses the plan for verifying the SIM astrometric performance requirement. Due to practical constraints, this effort involves a mixture of tests and modeling. The critical roles played by some models are circled, such as those needed to propagate the performance of each interferometer (single-baseline) to integrated system performance (3 baseline) or the incorporation of astrometric grid closure in assessing the ultimate 5 year mission accuracy.

Veteran analysts will admit that model validation is not straightforward in a quantitative, statistical sense and that much emphasis will likely be placed on fairly subjective, qualitative techniques[9]. Using a mix of model validation techniques in complementary fashion can reduce the risk of errors. Such techniques include[10]:

1) Face validation: inspection of model results by experts on the system being studied to confirm the model seems reasonable & provides the required functional capabilities
2) Peer review: independent review of the theoretical underpinnings and detailed examination of model internal components
3) Functional decomposition and test: also called *piece-wise* validation, inject test data into individual code-modules and compare actual output with predicted output
4) Comparison or empirical validation: compare performance of the model against performance with the physical system being modeled (or a similar system)

Comparison or empirical validation is a preferred technique and ideally provides quantitative estimates of model credibility or accuracy via cost functions such as Theil's Inequality Coefficient (TIC) and multivariate statistics[11,12].

In practice there are several limitations to this concept, chief among them: the final form of the system being modeled does not yet exist in the early phases of the project when the model is being used for requirements validation. This situation can often be remedied later in the project when the real system is undergoing test and can be used to support model evolution, although care must be taken to isolate noise induced by test artifacts from inherent system noise when making comparisons. In the early phase of the project, such model comparisons with real systems can sometimes be accomplished by modifying the model's capabilities to describe a similar existing system (thus validation by similarity).

A final word on Model Validation: a key to achieving model fidelity is to apply the proper rigor in designing the model. The aerospace community has embraced the concept of Test As You Fly (TAYF) on the verification front. We could likewise benefit from a analogous concept of Model As You Fly (MAYF), some basic guidelines for which are offered here:

1) Project should follow the "model as you fly; fly as you model" philosophy throughout its V&V activities. "Model as you fly" (MAYF) means:

- a. Mathematical models and simulations of physical processes and error sources (astrophysical and instrumental), hardware, and operational algorithms shall replicate to the maximum extent possible, the "real thing" (e.g., what the project system, environment, and algorithms will do in flight).
- b. These models and simulations should preserve spatial and temporal information and the same "causality" expected in flight (including order of occurrence)

2) MAYF is applicable to all "critical" models & simulations used to support an Analysis which serves as a primary Verification Method.

3) Exemptions to the MAYF philosophy, including characterization of differences between Modeled/Simulated and Flight conditions, should be documented in the MAYF Exemptions List and reported at Mission Readiness Review.

6. SYSTEM VALIDATION

Even if requirements and model validation result in a design that *should* meet the ultimate Need, and Verification proves that all formal requirements have been met, System Validation is necessary to prove the as-built system in fact *does* satisfy the ultimate Need.

System Validation goes beyond Verification – the primary intent is to assess robustness. When planning System Validation activities, the system engineer should be encouraged to *explicitly AVOID looking at the requirement documents* – instead think about "what needs to happen". While this may sound heretical, it's important to recognize that Validation is an organic rather than algorithm process and to pursue it accordingly.

Some primary System Validation methods include:

1. End-to-End Info System (EEIS) Testing
   - Addresses overall COMPATIBILITY of the Project information systems (command, data, timing, etc)

2. Mission Scenario Tests (MSTs)
   - Ensures flight h/w and all s/w can execute the mission under flight-like conditions (nominal & contingency) – but not on the flight time-line

3. Operations Readiness Tests (ORTs)
   - Demonstrates that all elements of the ground segment (software, hardware, people, procedures, & facilities) work together to accomplish the mission plan – i.e., using real timelines
   - Include Flight Segment for at least 1 or 2 of these and exercise key nominal and contingency operations

4. Stress-Testing (or Risk-Reduction Testing)
   - Demonstrates robustness to off-nominal conditions

5. Analysis & Simulation
   - Everything not covered by the above

To organize the System Validation program and ensure proper coverage, the System Engineer should develop and maintain a Validation Matrix which describes for each Key Functional Capability (usually by Mission Phase), how it is Validated – i.e., using which of the above methods, what "venue" (flight vehicle, testbed, simulator, etc). Note that this is similar to but independent of the traditional Verification Matrices – however, it is certainly appropriate to compare the two for purposes of eliminating unnecessary duplication.

A related concept is that of the *incompressible test list* which identifies the minimum set of tests that must be performed prior to certain project milestones such as launch, regardless of schedule and budget pressures. Once established, any changes to this "line in the sand" list must be approved by an unbiased, independent review authority (usually external to the project).

As mentioned previously, validation does not follow the same well-established policies as verification. When starting to put together a system validation matrix, the engineer may struggle with "how to start?". However, the same techniques discussed above in the section on Requirements Validation can be applied in generating such a matrix, which will define what validation tests are performed.

Validation should include Operational Readiness Tests (ORTs) that assess how the end-to-end system will really perform when all the flight and ground hardware, software, people, and operational procedures come together. Cross-system compatibility tests or "scrimmages" to validate things such as flight-ground interfaces are useful pre-cursors and complementary activities to ORTs. Another key component of system validation is stress testing and simulation, in which system robustness to variations in performance and fault conditions are assessed. The Region of Robust Operation in Figure 6 depicts the space over which system performance should be tested. Likewise, the results from system Fault Tree Analysis should guide the definition of Mission Scenario Tests, which include fault injection.

The importance of such stress testing and simulation was cited in the WIRE failure report: *"Testing only for correct functional behavior should be augmented with significant effort in testing for anomalous behavior"[3]*. Likewise from the MPL mishap report: *"The flight software was not subjected to complete fault-injection testing. System software testing must include stress testing and fault injection in a suitable simulation environment to determine the limits of capability and search for hidden flaws"[4]*. Some projects rightly claim they cannot do full fault-injection testing with the flight system without incurring excessive risks or costs. However, the solution is to invest in software testbeds to do this. Again, an early V&V Plan can identify the need for such testing and the required testbed capabilities.

Stress-testing validates the design by demonstrating system robustness by exercising over a broad parameter space – including beyond nominal. Some guildelines for planning stress-tests include:

1. Stress testing must be performed to determine capability boundaries and demonstrate

robustness of designs, in order to assure health/ safety and provide confidence in project validation.
2. Stress testing must consider single faults that cause multiple-fault symptoms, occurrence of subsequent faults in an already faulted state, etc (going beyond the traditional Fault Protection Verification program).
3. Stress testing will be performed on the integrated Flight Segment whenever it can be done safely and within resources. Otherwise it should be done in the System Test Bed.

Note that stressing in this context does NOT imply physically stressing the flight h/w. Planning such things also requires prioritization (balancing risk vs project resources) to determine which tests to run. Stress-test planning can benefit from a two-pronged approach.

- ORGANIC: Use brain-storming sessions with designers to identify what-ifs

- ALGORITHMIC: Use results of Analyses to identify "soft" spots
  - Fault-Tree Analysis
  - State Analysis
  - Performance Sensitivity Analysis

7. THE ROLE OF FAULT-TREES

The need for system-level risk analysis as part of the project validation program was highlighted in the findings of the 1998 MPL Mishap Investigation Board: "*A fault-tree analysis (FTA) was conducted by the project before launch for specific mechanisms and deployment systems where redundancy was not practical. No* ***system-level***[§] *FTA was formally conducted or documented…The greatest value of system-level FTA is to identify, from a top-down perspective, critical areas where redundancy (physical or functional) or additional fault protection is warranted*"[4].

Likewise, the MCO mishap investigation noted that a key contributor to that mission loss was: "*Absence of a process, such as fault tree analysis, for determining 'what could go wrong' during the mission"[13].*

Fault Tree Analysis (FTA) is a crucial tool for assessing the robustness of a system to failure modes. An FTA is simply an exercise in which the system engineer considers "what has to happen for the mission to succeed" (or conversely, fail) and then de-composes this in a logical fashion. This is very useful in later phases of system validation, particularly test planning.

It is fairly standard practice for projects to employ some form of FTA to assess reliability of individual mechanical actuators. Unfortunately, the use of FTA to study overall system-level robustness is used less consistently. Recent projects such as Mars Odyssey have benefited from the MCO/MPL lessons-learned and implemented rigorous FTA and Risk Assessment programs, guidelines for which have been well documented in the literature[14].

In addition to system-level FTA, lower-level design risk analyses are necessary to ensure integrated system robustness. A companion to FTA is the Failure Modes, Effects, and Criticality Analysis (FMECA). A cognizant engineer for an assembly is responsible for insuring by analysis that failures in that assembly cannot propagate to other assemblies (e.g., a short-circuit condition). FMECA therefore tends to be focused on the interfaces between project elements. Additionally, *Functional* FMECAs can be performed to identify failure modes associated with key functions (as opposed to a device- or interface-centric view). FMECAs, when combined properly with system FTA, help ensure the objective of *fault containment* is met.

It should be stressed that Risk Analysis in general and FTA in particular are activities that span the project life-cycle. They should begin in Definition phase, become particularly active around the time of Critical Design Review, and continue into the Operations phase and planning for operations (in terms of contingency planning). This "healthy questioning of what might go wrong" is frequently cited as a good feature of a successful project culture[10]. However, in order for Risk Analysis to be effective, the person responsible for leading this effort must have sufficient time, broad perspective, and management-supported clout to keep a watchful eye across the project and ferret out risks before they develop into unrecoverable problems. This person is a key link in the project's risk-avoidance chain and having a "Sherlock Holmes approach and a bulldog's disposition to pursue strange indications while the rest of the team is distracted" (well into operations) is vital[8]. One final point about FTA, this activity is ideally done independently – even if it involves project personnel involved in requirements synthesis. Taking a fresh look from another angle serves as a form of *independent* requirements validation.

Some guidelines for Mission Fault Tree Analysis include:

- Mission-level FTA starts with a Mission Success Tree which is inverted to form Mission Level Fault Trees by Mission Phase
- Mission-level FTA should "come down" and meet subsystem-level Fault Trees "coming up"
- Project Systems Engineering is responsible for Mission-level FTA, delegated to the Fault Protection Team who also does the subsystem level FTA (with support of the subsystem designers)

[§] Emphasis mine.

- Mission-level FTA assists with Stress-testing/validation by identifying specific failure modes

A final use of Fault Trees Analysis is identifying soft-spots in actual test design. Particularly complex and critical tests should have fault-trees developed to clearly identify "where this can go wrong" and add appropriate mitigating features.

8. SUMMARY

In conclusion system engineers should keep the following points in mind when validating a complex system:

1. Validation is not the same as Verification
2. Validation occurs across the project life-cycle
3. Projects (& Validation efforts) are 3-dimensional
4. Requirements Validation involves 5 questions
5. Use storyboards to identify critical models
6. Model As You Fly (MAYF)
7. Performance Robustness → Merit Functions
8. Functional Robustness → State Analysis & Stress-Testing
9. Recognize & Manage Uncertainty Factors for Testing & Models
10. Identify & Correct soft spots → Fault Trees

9. ACKNOWLEDGEMENTS

The author is grateful for useful discussions with colleagues including: Arden Acord, Mark Brown, Ross Jones, Gentry Lee, Charles Whetsel, Tom Gavin, and Bob Rasmussen at JPL, Steve Jolly at Lockheed Martin Astronautics, and Eric Bachtell at Ball Aerospace and Technology Corporation. The MER Terminal Descent fault tree was provided by Bob Mitcheltree of JPL and NASA Langley. Kepler Principal Investigator William Borucki of NASA's Ames Research Center developed the Science Merit Function described here. Mike Margulis of Lockheed Martin Missiles & Space Systems and JPL's Peter Kahn provided input on the SIM astrometric performance validation flow.

The work described here was carried out at the Jet Propulsion Laboratory, California Institute of Technology under contract to the National Aeronautics and Space Administration.

10. REFERENCES

[1] Gehman, Hal, etal, "Columbia Accident Investigation Board report", Vol I, URL: http://caib.nasa.gov/news/report/volume1/chapters.html, August 2003, [cited Nov 1, 2005]

[2] Bradley, Theron, etal "CONTOUR Mishap Investigation Board Report", URL: http://www.nasa.gov/pdf/52352main_contour.pdf, May 31, 2003 [cited Nov 1, 2005].

[3]Branscome, Darrell R., "WIRE Mishap Investigation Board Report", URL: http://klabs.org/richcontent/Reports/wiremishap.htm, June 8, 1999 [cited August 3, 2003].

[4]Casani, John, "Report on the Loss of the Mars Polar Lander and Deep Space 2 Missions", URL: http://spaceflight.nasa.gov/spacenews/releases/2000/mpl/mpl_report_1.pdf, March 22, 2000 [cited August 8, 2003].

[5]Grady, Jeffrey O., *System Validation & Verification*, pp. 74-75, CRC Press, 1998.

[6] Hoyle, David, *ISO9000 Quality Systems Handbook*, p.234, 3rd edition, 1998.

[7]Hoban, Frank, Hoffman, Ed, "NASA Systems Engineering Handbook", NASA SP-610S, June 1995

[8]Euler, Edward, Jolly, Steven, Curtis, H.H., "The Failures of the Mars Climate Orbiter and Mars Polar Lander: A Perspective From the People Involved", Vol 107, *Advances in the Aeronautical Sciences*, American Astronomical Society, 2001.

[9]Arthur, James D, Sargent, Robert G. , Dabney, James B., Law, Averill M., and Morrison, John D., "Verification and Validation: What Impact Should Project Size and Complexity have on Attendant V&V Activities and Supporting Infrastructure", *Proceedings of the IEEE: 1999 Winter Simulation Conference*, Institute of Electrical and Electronic Engineers, edited by P.A. Farrington, pp. 148-155, 1999.

[10]Youngblood, S.M., Pace, D.K., "An overview of model and simulation verification, validation, and accreditation", *Johns Hopkins APL Technical Digest*, 16(2): 197-206, Apr-Jun 1995.

[11]Smith, M.I., Hickman D., Murray-Smith, D.J., "Test, verification, and validation issues in modelling a generic electro-optic system", *Proceedings of the SPIE: Infrared Technology and Applications XXIV*, Vol 3436, pp. 903-914, July 1998.

[12]Balci, O., "Principles of simulation, model validation, verification, and testing", *Transactions of the Society for Computer Simulation International*, 14(1): 3-12, March 1997.

[13]Stephenson, Arthur G.,"Report on Project Management Within NASA by the Mars Climate Orbiter Mishap Investigation Board", URL: http://www.space.com/media/mco_report.pdf, March 13, 2000 [cited August 8, 2003].

[14]Beutelschies, Guy, "That One's Gotta Work – Mars Odyssey's use of a Fault Tree Driven Risk Assessment Process", *Proceedings of the IEEE Aerospace Conference*, 6599-2, 2001.

BIOGRAPHY

Riley Duren is a Senior staff member at the Jet Propulsion Laboratory (JPL). He is currently the Project System Engineer for the *Kepler* space observatory, scheduled for launch in 2008 and designed to determine the frequency of extra-solar terrestrial planets. He has held systems engineering positions on other projects associated with exo-planet detection, including the *Starlight* mission (a formation-flying interferometer precursor to the *Terrestrial Planet Finder* mission) and the *Space Interferometry Mission*. He was the Chief Engineer for the Attitude and Orbit Determination Avionics and Mission Operations Director for the *Shuttle Radar Topography Mission,* which in 2000 generated a near-global map of the earth's surface to 10 meter vertical accuracy. He has also served on the Mars Program Systems Engineering Team and is a consultant on metrology systems for large aerospace structures. Before joining JPL, he was a lead Test Engineer on five space shuttle science payloads at NASA's Kennedy Space Center (including tethered satellites, condensed matter physics, astronomy, and remote sensing).